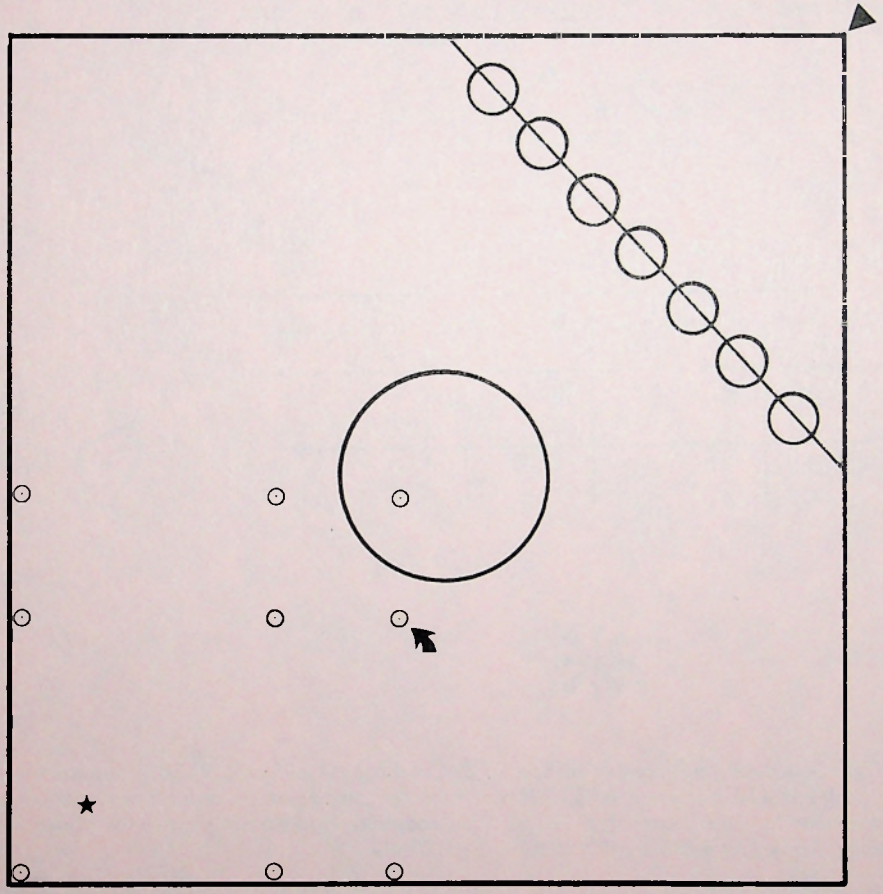
VOL 3 NO 2 FEBRUARY 1975

23

Popular Computing



SPACESHIP

In this issue: Numerical Analysis vs Numerical Methods // Penny Flipping Problems

A spaceship is located at the point marked with a star and is attempting to reach the upper right corner, marked with a triangle. For convenience, the area has been scaled on one parsec, forming a unit square. The spaceship is located at (.100,.100). Its movements are constrained two ways:

1) Its coordinates may be altered by

squaring	(S)
cubing	(C)
square rooting	(2)
cube rooting	(3)

2) It may not enter or cross some danger areas, indicated by the circles. The large circle has a diameter of .25 and is centered at (.500,.500); the small circles each have diameters of .075 and are evenly spaced along the line Y = -X + 1.5.

At any stage, then, the ship has 16 possible moves. From its initial position, it may move to one of these 16 positions:

S .010 C .001		S .010 S .010		S .010 2 .316		S .010 3 .464	
C .001 C .001		C .001 S .010		C .001 2 .316		C .001 3 .464	
2 .316 C .001		2 .316 S .010		2 .316 2 .316		2 .316 3 .464	
3 .464 C .001		3 .464 S .010		3 .464 2 .316		3 .464 3 .464	

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $18 per year, or $15 if remittance accompanies the order. For Canada and Mexico, add $4 per year to the above rates. For all other countries, add $6 per year to the above rates. Back issues $2 each. Copyright 1975 by POPULAR COMPUTING.

Publisher: Fred Gruenberger
Editor: Audrey Gruenberger
Associate Editors: David Babcock, Irwin Greenwald
Contributing editors: Richard Andree, Daniel D. McCracken, William C. McGee
Advertising manager: Ken W. Sims
Art Director: John G. Scott
Business Manager: Ben Moore

nine of which are indicated by tiny circles. For only four of the 16 possible moves is the ship proceeding toward its goal, and one of those (.464,.464) lies in a danger area. Suppose that the captain decides to apply the transformation $\sqrt[3]{X}, \sqrt{Y}$ and move his ship to (.464,.316), to the point indicated by the arrow. For his next move, then, he again has 16 choices:

S .215 C .032		S .215 S .100		S .215 2 .562		S .215 3 .681	
C .100 C .032		C .100 S .100		C .100 2 .562		C .100 3 .681	
2 .681 C .032		2 .681 S .100		2 .681 2 .562		2 .681 3 .681	
3 .774 C .032		3 .774 S .100		3 .774 2 .562		3 .774 3 .681	

and so on. He must continue to choose moves that will guide the ship toward the goal without entering or crossing the danger areas; that is, the straight lines that connect his move points must not intersect the danger circles. When both coordinates exceed .99, the goal will be reached.

PROBLEM 76

(A) What choices should the captain make?

(B) What is the smallest number of moves that will get the spaceship to its goal?

(C) What is the shortest distance to the goal? (The length of a side of the unit square shown is 19.15×10^{12} miles).

Numerical Analysis vs Numerical Methods

Numerical Analysis (NA) is a traditional subject in mathematics. It deals with the theory and algorithms of such topics as: function evaluation; error analysis; finite differences; roots of equations; roots of systems of equations; numerical integration; numerical solution of differential equations; calculation of eigenvalues. As presented by mathematicians, the subject involves derivations, existence proofs, closed-form solutions, and analytic expressions of error bounds.

Numerical Methods (NM) is a subject in computing. It covers the same topics listed above, but from an entirely different point of view; namely, that of coercing answers from a computer (or from a desk calculator) and providing some means of validating those answers. In addition to the traditional topics, a Numerical Methods course may include many other topics of computational interest, such as factor analysis, chi-squared analysis, curve fitting, and the analysis of parametric equations.

Since the heart of the study of NM is usually the computer, the differences between the two subjects are profound. To begin with, mathematicians operate in a world in which all numbers exist. In the world of computers, most numbers do not exist. Numbers like pi and e obviously do not exist, but neither do simple numbers like 1/7. In single-precision integer work on a 32-bit machine, there are exactly 2^{32} numbers all told. In 8-digit decimal scientific notation (i.e., normal Fortran) with an exponent range of $\pm$ 100, there are about $18 \cdot 10^9$ numbers all told (the high order digit of the mantissa is not allowed to be zero. Since the internal representation is probably binary or hexadecimal, the actual number of numbers is slightly different). The range can be extended through multiple precision, but no matter how far the precision is extended, there is always a finite limit (and the number 1/7 still does not exist). Thus, comforting assumptions that can be made in NA, such as

$$A \cdot (1/A) = 1$$
$$(\sqrt{B})^2 = B$$
$$(A + B) + C = A + (B + C)$$

are not valid in computing. Further, if scientific notation is used, as it usually is, the number system is not uniformly dense; that is, the packing of numbers near zero is tighter than at each decade removed from zero. At the far limits of the so-called floating point system, using 8 significant digits and a range of $\pm$100 on the exponent, the closest that two successive numbers can be to each other is 10^{92}. (If the smallest distance between successive numbers near zero is expressed as a micrometer, then the distance between successive numbers at the limit of the system is around 10^{75} light-years). This non-homogeneity produces some odd and unpredictable effects on what would otherwise be simple calculations.

In mathematics, formulas like

$$\ln(1+z) = z - (1/2)z^2 + (1/3)z^3 - \cdots$$

can be invoked; it is those three dots at the end that mark a sharp line of ditinction with Numerical Methods. Not only is it not possible to carry out an infinite calculation, but the formulas are somewhat misleading, since they may call for extended calculations at parts of their range that are not feasible. In the world of NM, we deal with finite machines and also finite amounts of time. The harmonic series,

$$1 + 1/2 + 1/3 + 1/4 + 1/5 + \cdots + 1/k$$

is divergent mathematically; in a world of finite arithmetic it converges (see PC9-14).

In NA, the concept of continuity leads to techniques that aid in problem solution. In NM, due to the discrete nature of all numbers, the concept of continuity may be missing. In NA, a matrix is either singular or non-singular; in NM, there can be "almost singular" matrices. Division by zero is outlawed in both fields, but in NM division by "nearly zero" is also dangerous, and usually more so (since zero is easily tested for, but "nearly zero" is not).

There have been hundreds of books in NA, but few, to date, in NM. Most books which pose as NM books are simply NA books with some passing references to computers, or to Fortran. An outstanding exception is the book by Forman Acton, Numerical Methods That Work*, which is pure NM from page 1 on, and pure gold throughout. Some of the flavor of Acton's book comes through in his discussion of a problem posed by Wilkinson**. Take the polynomial

$$(x+1)(x+2)(x+3)(x+4)\cdots(x+19)(x+20) = 0$$

This equation has 20 real roots, nicely defined. The expanded form of the polynomial is:

$$x^{20} + 210x^{19} + 20615x^{18} + \cdots + 2432902008176640000 = 0.$$

Wilkinson changes the coefficient of x^{19} by adding 2^{-23}. This changes the 210 to 210.000000119209289550781 25 (a change of one unit in the 10th significant digit). To quote Acton, "Our intuition suggests that such a slight change could not seriously affect the roots of the polynomial, but our intuition is wrong!" Accurate calculation of the roots shows them to be:

---

*Harper & Row, 1970. The title quoted is what appears on the title page, but the cover has added to it the word "usually," pressed into the cloth.

**"The evaluation of the zeros of ill-conditioned polynomials," Numerische Mathematik, 1 (1959), pages 150-166.

-0.99999 99999 99999 99999 99990
-2.00000 00000 00000 00976 20
-2.99999 99999 99805 23297 6
-4.00000 00002 61023 18914
-4.99999 99275 51537 90956
-6.00000 69439 52295 70720
-6.99969 72339 36013 949
-8.00726 76034 50376 855
-8.91725 02485 17070 494
-20.84690 81014 82

-10.09527 ± 0.643501
-11.79363 ± 1.652331
-13.99236 ± 2.518831
-16.73074 ± 2.812621
-19.50244 ± 1.940331

"We see that serious changes have appeared in all the roots from -9 to -20, with 10 of them becoming complex. Not even slightly complex, either; the imaginary parts are large. All these changes, it must be emphasized, are real. They are not mistakes in computing the roots; rather they are actual changes in the roots of our polynomial produced by changing one of its coefficients by one in the 10th significant figure. If we must find the roots of this polynomial, we must certainly perform all our arithmetic with an accuracy greater than ten significant figures, no matter what algorithm we use, and our algorithm must not let us stop before each root is determined to an accuracy sufficient not to perturb the further roots we have still to find.

"This example is horrifying indeed. For if we have actually seen one tiger, is not the jungle immediately filled with tigers, and who knows where the next one lurks?"

Acton's book is a constant delight. There is something worth quoting on almost every page. Some samples are given here:

"On an automatic computer, the need to switch all arithmetic to complex numbers on the next cycle of an iteration is a blasted nuisance, though certainly possible. In hand computation, the aggravation is possibly less."

"Since the basic question-answering routine is not bothered by multiple roots, it would seem that this iterative procedure might be the universal polynomial root finder that we seek. Alas no! Tests on a number of polynomials, both nasty and nice, show that Lehmer finds roots about as precisely as one of the unattractive standard packages but takes three times as long. (The package includes a stepping search leading to Newton in one dimension for the real roots, Bairstow for quadratic factors, and a final Newton polishing.) Thus we are faced with one of those unpleasant triumphs of the efficiency of brute force. We can only take solace in that they seem to be rare."

"A system cannot do everything: There will always be some extremely nasty polynomials which will frustrate it and cause it to fail to find some of the roots."

"The student who worships at the altars of Classical Mathematics should really be warned that his rites frequently have quite oblique connections with the external world."

"At a somewhat trickier level, the person who wrote, but did not clearly document, a vector output subroutine so that it needed the address of the vector not the number of items to be put forth was quite mystified when irate customers who wanted 10 or 12 items were swamped with 24371 of them--their number of items, N, being stored typically in location 24371. Fortran, it must occasionally be remembered, always passes addresses to subroutines regardless of what its notation may imply to the casual observer. This last example is merely one of hundreds of troubles that are produced by programmers who are insensitive to the interests of the users of their programs. They fail to match properly their program impedances with probable human inputs."

"Having hinted darkly at my computational fundamentalism, it is probably time to commit a public heresy by denouncing recursive calculations. I have never seen a numerical problem arising from the physical world that was best calculated by a recursive subroutine--that is, by a subroutine that called itself. I admit the idea is cute and, once mastered, it tends to impel its owner to apply it wherever possible--all questions of appropriateness aside."

"Whenever a person eagerly inquires if my computer can solve a set of 300 equations in 300 unknowns, I must quickly suppress the temptation to retort, 'Yes, but why bother?'"

Probably the biggest difference between NA and NM is found in the computer's ability to do dog work. In a sense, NM admits of some sloppiness (although we usually wind up trying to squeeze one more digit out of our calculations). For example, take the notion of rounding. At one time, rounding was a statistical technique, as seen in the 1948 edition of Snedecor's Statistical Methods:

"In rounding, numbers like 13.51 are increased to 14, while such as 13.49 are lowered to 13. The dividing line being 13.50, what shall be done if this particular number appears? In order to insure increases and decreases in approximately equal proportions, a good rule is to round to the nearest even number. Thus, 13.50 and 14.50 would each be rounded to 14."

With the advent of computers, the rule became much simpler: add 5 in the position to be dropped. The theory is that with computers, one can always arrange to carry more places--precisely the dog work principle mentioned above.

Since NM is a computing subject, it reduces to a course in problem solving with computers. The problems to be solved happen to be the fundamental problems of NA, but subject to the power of the computer and to its limitations and booby traps. The solution guaranteed us (perhaps in closed form) in NA must first of all be programmed, and that immediately introduces the complexities and awkwardnesses of whatever programming language we use. Then that program must be compiled. Robert Teague's column, "Speaking of Languages..." points out repeatedly the hazards of that process. And even assuming that compilation proceeds smoothly, there are awesome tigers waiting to pounce when finite, non-homogeneous arithmetic is applied to what should be a straightforward procedure.

Anyone who has taught computing has had the experience of receiving a student's project report, frequently representing a semester's work, that displays as output what is patently garbage of the worst sort. The student involved has apparently not heard a single word of caution about the necessity of validating his results, and proudly presents his nonsense as truth (after all, the printing did come from a computer). He has probably made one of the following blunders:

1. He has utilized a packaged routine in a way not anticipated by its author, such as invoking the package outside its normal range.

2. He has totally perverted the theory of a package. An example of this is coding a random number generator in floating point, where the whole theory demands integer arithmetic.

3. He has used a technique that works in theory (such as a series calculation) but fails in practice, due to the finite nature of the computer's processes.

4. He has used a brute force method (e.g., solving a combinatorial problem by exhaustion) and wasted hours of CPU time on less than 1% of the combinations, but "reasoned" that the hours of time must, somehow, have made a dent in the exploration of the solution space.

5. He has obtained correct results but failed to display them properly, due to obvious misuse of FORMAT statements in the output.

If the two subjects are indeed different and distinct, even though they seem to cover much the same ground, then the student of computing would be well advised to take both of them, provided that both are offered. If only one course is offered, then the differences and distinctions should be pointed out at every place that they occur.

Incidentally, Wilkinson's equation provides fertile ground for further work in a semester term project. The 21 coefficients of the polynomial are as follows:

```
                                1.
                  210.00000 01192 09289 55078 125
                       20615.
                   12 56850.
                  533 27946.
                16722 80820.
              4 01717 71630.
             75 61111 84500.
           1131 02769 95381.
          13558 51828 99530.
        1 30753 50105 40395.
       10 14229 98655 11450.
       63 03081 20992 94896.
      311 33364 31613 90640.
     1206 64780 37803 73360.
     3599 97951 79476 07200.
     8037 81182 26450 51776.
    12870 93124 51509 88800.
    13803 75975 36407 04000.
     8752 94803 67616 00000.
     2432 90200 81766 40000.
```

N-SERIES 23

Log 23	1.36172783601759287886777711225118954969751103433609 6
Ln 23	3.13549421592914969080675283181019611844238031484043 6
$\sqrt{23}$	4.79583152331271954159743806416269391999670704190412 9
$\sqrt[3]{23}$	2.84386697985156547769543940095865185276416517273704 8
$\sqrt[5]{23}$	1.87217123055485741669565788145194320097286646708792 9
$\sqrt[7]{23}$	1.56506560796023959512405515446087216098691263514509 9
$\sqrt[10]{23}$	1.36827308332615292176630582488599799235174911915679 4
$\sqrt[100]{23}$	1.03185168656154033089285447901518827247460686975422 2
e^{23}	9744803446.248902600034632684822975277649387764036006 976355903227801470169706381193194790280416
π^{23}	271923706893.615930060980328223255795659076537028384 1 271464495493078769867685802682971265578 9
$\tan^{-1}$ 23	1.52734543140336577715814734751375988059110001970544 8
23^{100}	14886191506363039393791556586559754231987119653801368 68657698820922243327853933135215239014327734680423347 65921794473108595202225298760 01

The Penny Flipping Problems

In the book SIMULA BEGIN (Auerbach Publishers, 1973), the Penny Flipping Problem is given as an illustration of the power of the SIMULA language. The problem is attributed to Iain Bride and John Gilder of the University of Manchester Institute of Science and Technology, and is as follows:

A pile of N pennies is arranged so that each penny is heads up. We define an operation FLIP(M) on this pile, which removes the top sub-pile of M pennies, turns the sub-pile upside down and replaces them on top of N - M pennies remaining. The consecutive operations

FLIP(1),FLIP(2),...,FLIP(N),FLIP(1),FLIP(2),...

are repeated until the stack returns to all heads. The value of F for a given N is the number of flips required to return the stack to heads. For N = 6, the 35 flips are shown in this list, where 0 stands for heads and 1 for tails.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
0	1	1	1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	0	1	1	0	1	0	1	1	0	0	1	1	0	1	0	1	0	
0	0	0	1	1	1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	0	1	1	0	0	0	1	1	0	0	0	1	0	1	0	
0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	1	0	1	1	1	0	0	1	1	1	1	0	1	0	
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	0	0	0	1	0	
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	

PC23-10

We shall refer to the original problem as PF-I. The functional values for stacks of coins up to 32 are given in the accompanying table. (See page 12)

The original Penny Flipping Problem is already famous, but more for its value in demonstrating the elegance of SIMULA than its interest as a problem. No theoretical evidence has been presented to conclude that the stack will return to all heads. The number of permutations of N objects is limited, of course, but there remains the possibility of cycling among the (N!) permutations, with the cycle not including the winning arrangement. Notice that the problem is not serial in nature; that is, one can seek the value of F(100) without having found any prior values. Thus, many people can work independently on extending the function.

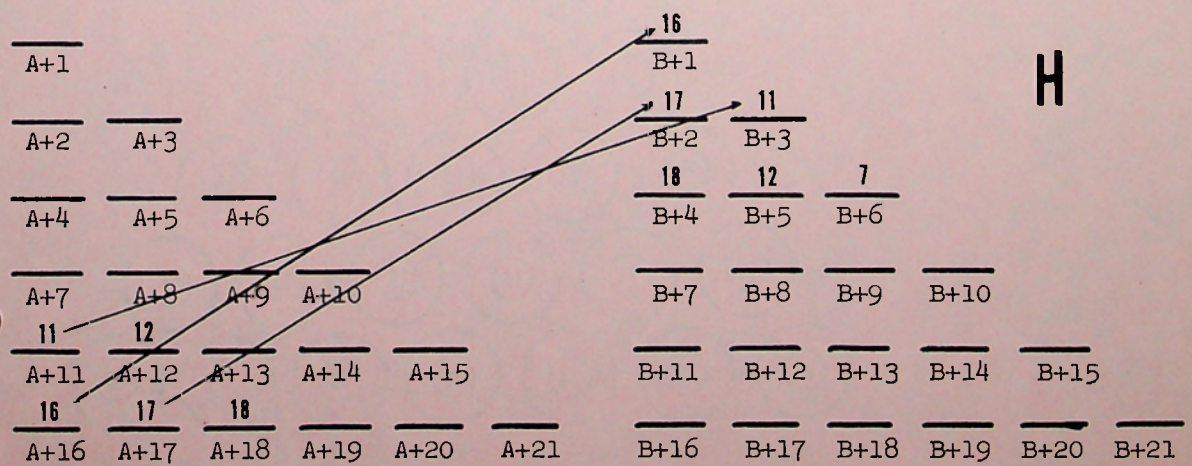
The problem can be made more interesting in the following way. Turn over M pennies at each flip, but alternately from the top and bottom of the pile of N. The following list shows the 36 flips for N = 6 in this second version, called PF-II. The value of the function is similar to that of the original problem for small N, but becomes larger as N increases.

0	1	1	1	1	0	0	1	1	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	0	1	1	1	1	1	0	1	1	0	0	1	0
0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	0
0	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	1	0	1	1	1	0	1	0	0	0	0	0	1	0	1	1	1	0	0	1	0
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	0	0	1	1	1	1	0
0	0	1	1	1	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	0	0	0	0	0	1	1	0	0	1	1	1	1	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	

Entries in the table of values for PF-II are solicited from readers. Preliminary work indicates that the functional values for PF-II may be quite large; for example, for a stack of 44 coins, 230384 flips are needed. Those who wish to extend this research are urged to pick values of N above 50, so that the work of different people will extend the knowledge of the function.

A third problem (PF-III) suggests itself. Consider a triangular array of pennies, as shown. Starting as before with all heads up, the play proceeds as follows. Starting at the top (at the circle labelled 1), flip one row, then two rows,..., then all of the array. Now work from the lower left corner (the circle labelled 22), and flip successive rows. Then work from the lower right corner (the circle labelled 28), flipping one, two,..., all rows.

For computer work on PF-III, the problem is simplified by inclusion of a subroutine to rotate the array in storage. See Figure H. Given the array of numbers in storage, the contents of the array is to be moved to a second array to perform the rotation (120° clockwise). The second array is then moved back to the original storage locations, so that the instructions for flipping can operate on the same words. PF-III itself is probably not as interesting a problem as PF-II, but this sub-problem of rotating an array of numbers in storage is a delightful exercise in computer logic.

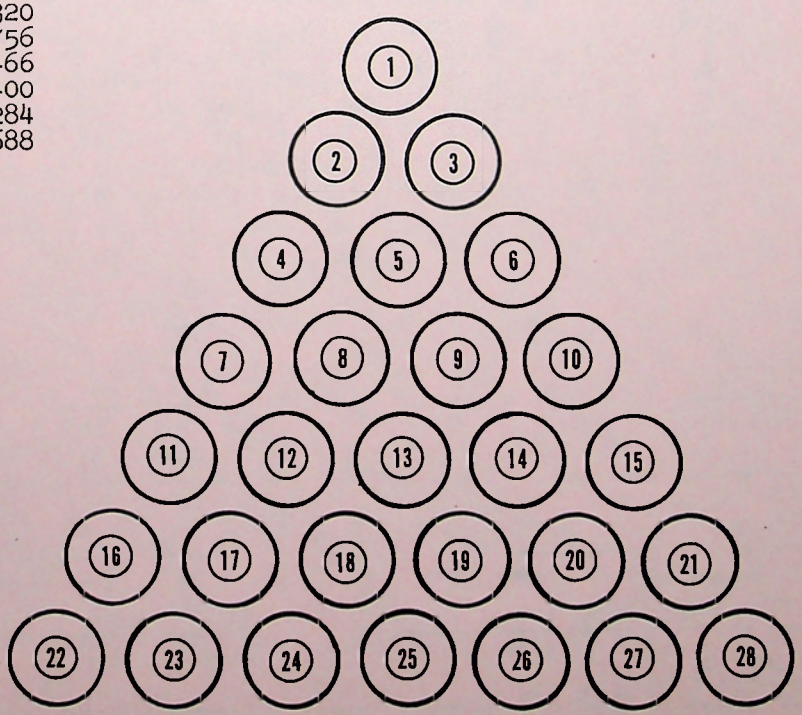


H

Known results for the three Penny Flipping Problems

N	I	II	III
1	2	2	2
2	3	3	6
3	9	3	8
4	11	11	12
5	24	20	
6	35	36	
7	28	70	
8	31	55	
9	80	80	
10	60	295	
11	121	121	
12	119	119	
13	116	130	
14	195	1176	
15	75	300	
16	79	1440	
17	204	1428	
18	323	576	
19	228	2128	
20	199	320	
21	146	440	
22	264	483	
23	529	2622	
24	504	432	
25	200	1900	
26	675	780	
27	540	4320	
28	251	756	
29	840	4466	
30	899	26400	
31	186	11284	
32	191	2688	

The functional values for PF-II were calculated by Thomas Sardi.



To make sure that the rules for PF-III are clear, for the case N = 7 shown in the illustration, the successive flips would involve these coins:

```
 1
 1  2  3
 1  2  3  4  5  6
 1  2  3  4  5  6  7  8  9  10
 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
 1  2  3  4  5  6  .........19  20  21
 1  2  3    (the entire array)      26  27  28
22
22 16 23
22 16 23 11 17 24
22 16 23 11 17 24  7  12  18  25
22 16 23 11 17 24  ....  18  25  4  8  13  19  26
22 16 23 11 17 24  ....   2   5  9 14  20  27
22 16 23    (the entire array)      15  21  28
28
28 21 27
28 21 27 15 20 26
28 21 27 15 20 26 10 14 19 25
28 21 27 15 20 26 10 14 19 25  6  9  13 18  24
28 21 27 15 20 26 10 ....  3  5  8  12 17  23
28 21 27 15   (the entire array)
 1
 2  3
 4  5  6  etc.
```

Almost Primes Plus Two

PROBLEM 79

What is the probability that the number AB+2, where A and B are odd primes, is prime?

131 · 139 = 18209	18211 is prime
149 · 151 = 22499	22501 is prime
151 · 157 = 23707	23709 is composite.

A short computer run was made by Dorothy Cady, using the primes between 101 and 2003 for both A and B, indicating that the answer is around .10; that is, about nine times out of ten, AB+2 is <u>not</u> prime.

Further research into this problem is needed, either with larger values for A and B, or with many more values; or there may be an analytic solution.

PROBLEM 80

An old problem (E2190) in the American Mathematical Monthly called for a proof of this statement:

If N and M are positive integers, then the smaller of

$$\sqrt[M]{N} \text{ and } \sqrt[N]{M}$$

cannot exceed $\sqrt[3]{3}$. This appears to be true.

If we now add the constraints that $N \neq M$ and that both N and M be less than 1000000, then there is a computational problem; namely, how close can we get to $\sqrt[3]{3}$?

Common Data

PROBLEM 81

Four blocks of storage each contain 1000 numbers. Write a program to count how many of the numbers:

(A) Appear in all four blocks.
(B) Appear in any three of the blocks.
(C) Appear in any two of the blocks.
(D) Appear in only one block.

Indicate how this program might be tested.

In PC21-9, the calculation of e was shown as applied directly to the Taylor series expansion of e^x.

Dr. John Wrench points out that his calculation, with Daniel Shanks, in 1961 went to 100265 digits. He also points out a different algorithm, the one used by D. J. Wheeler in 1953:

$$a_k = (1 + a_{k-1})/(n - k)$$

$$a_0 = 1/n$$

which converges to e-1 after n-1 steps with an error of the order of 1/n! For n = 15, for example,

$$a_0 = .06666666666667$$

$$a_1 = (1/14)(1.06666666666667) = .07619047619047$$

$$a_2 = (1/13)(1.07619047619047) = .08278388278388$$

.

.

$$a_{14} = 1.71828182845899$$

Terms in Computing for the Deaf

California State University, Northridge, has produced four films of American Sign Language of terms in computing and data processing, for use by deaf students. Each film contains 30 terms, taken generally in the order of frequency of use in a beginning course in computing; that is, film No. 1 contains the most-used terms.

The films are in color and run at sound speed (24 frames per second), although there is no sound track. At this speed, each film runs approximately six minutes. The films can be run at silent speed; the screen time of each film is then approximately nine minutes.

A term, such as "computer," is presented to the viewer three times. First, the term is spelled out. Then the sign for the term is shown, seen from the front as the recipient would see it. At this time, the term also appears on the screen in English letters, and the signer is also seen saying the term. At the third appearance, the sign is made again, but seen from the rear as the user would view it, and it again appears in English letters.

The films, made with the cooperation of CSUN's Center on Deafness (Ray Jones, Director), will not be distributed by the University. Interested groups, on request, will be authorized to purchase prints directly from the film laboratory, at the laboratory's cost. The price for each film is different (due to slightly different lengths) and the price will change with time, but roughly each print will cost $28 plus shipping; this price includes a reel, can, and mailing case.

The films were made under a grant from IBM, and were produced by Fred Gruenberger. The choice of terms, and the creation of the hand signs, was done by Robert Teague and Carl Kirchner; Mr. Kirchner appears on the screen as the signer. Direction and editing were done by John Gruenberger, a member of CSUN's Department of Art.

For further information, contact Prof. Fred Gruenberger at California State University, Northridge 91324, phone (213) 885 2439.

Some frames from the actual films are included with the initial print run of this issue.

PC23-16

ANNUAL INDEX

Volume 2, 1974 Issues 10 through 21/ 192 pages